Actividad 05

Alumno: José Osvaldo Farfán de León

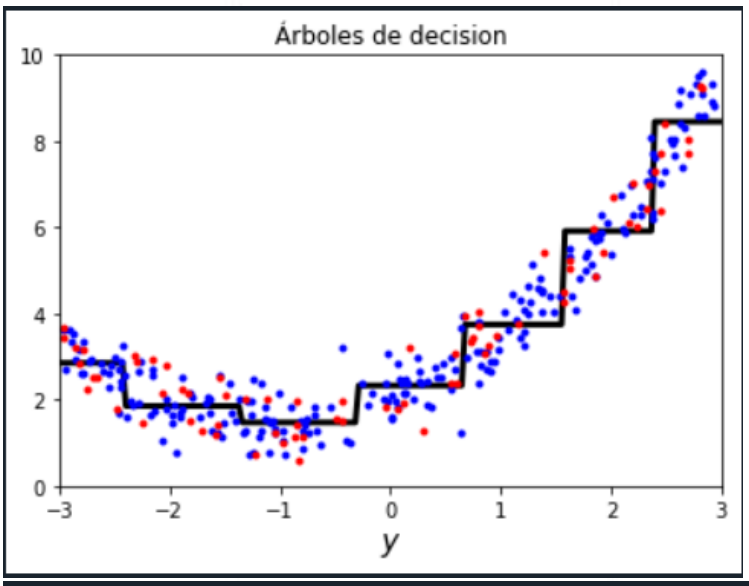
Materia: Seminario de IA ll

Profesor: Carlos Alberto Villaseñor Padilla

Sección: “D04”

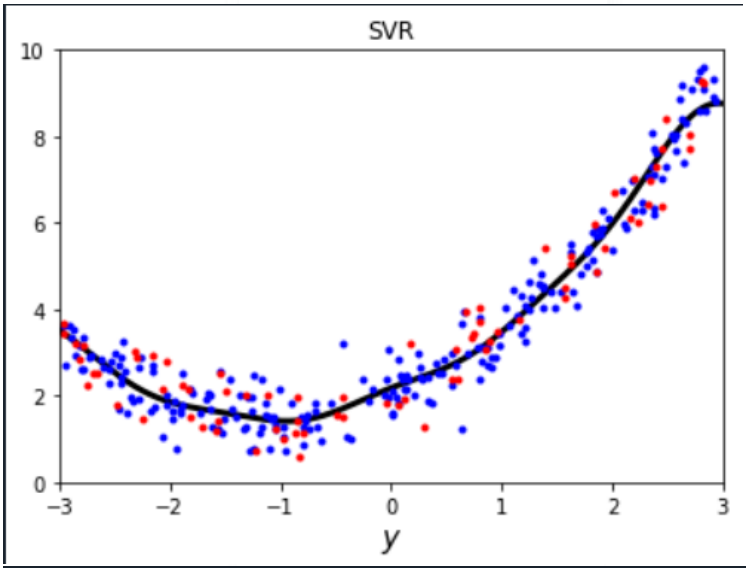
Comparación de Regresores no lineales

Decision Tree



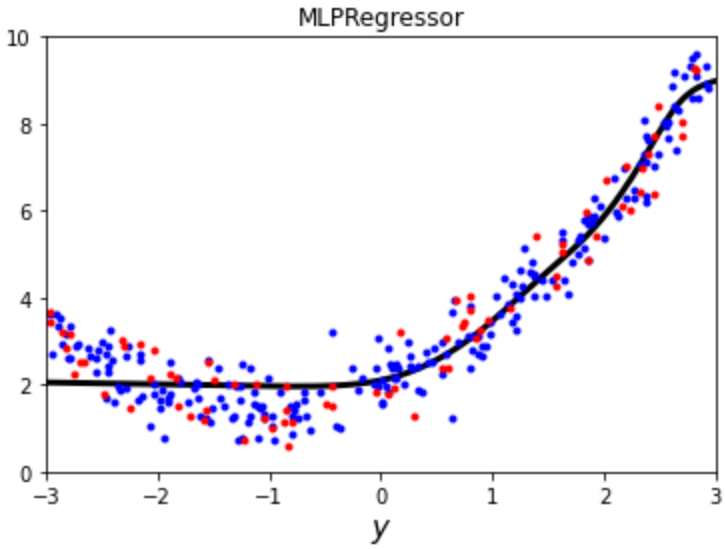

```
In [2]: runfile('/home/farfan/.config/spyder-py3/temp.py', wdir='/home/farfan/.config/spyder-py3')
Train:  0.9552085891810569
Test:  0.9372882191359417
```

SVR



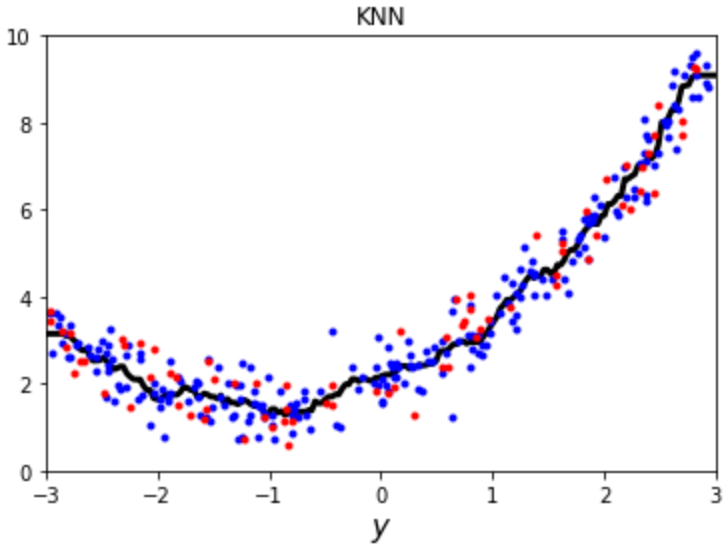

```
In [4]: runfile('/home/farfan/.config/spyder-py3/temp.py', wdir='/home/farfan/.config/spyder-py3')
Train:  0.9552085891810569
Test:  0.9372882191359417
```

MLP



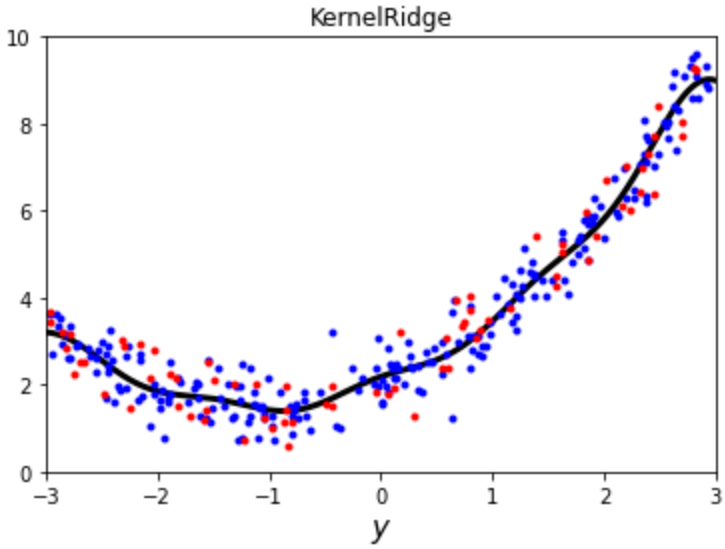

```
In [5]: runfile('/home/farfan/.config/spyder-py3/temp.py', wdir='/home/farfan/.config/spyder-py3')
/home/farfan/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/_multilayer_perceptron.py:1625: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Train:  0.9316408802649183
Test:  0.9067144812538684
```

KNN




```
In [6]: runfile('/home/farfan/.config/spyder-py3/temp.py', wdir='/home/farfan/.config/spyder-py3')
Train:  0.959008077212043
Test:  0.9373884577520193
```

Modelo Kernel Ridge




```
In [7]: runfile('/home/farfan/.config/spyder-py3/temp.py', wdir='/home/farfan/.config/spyder-py3')
Train:  0.9573111036998189
Test:  0.9373459499797254
```

Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

#Crear datos sinteticos
np.random.seed(42)
m = 300
ruido = 0.5
x = 6 * np.random.rand(m,1) - 3
y = 0.5*x**2+x+2 + ruido * np.random.randn(m,1)
xtrain, xtest, ytrain, ytest = train_test_split(x, y)
```

```
#Crear y entrenar modelo
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor(min_samples_leaf = 20)#entre mas bajo tiende a sobreentrenarse
model.fit(xtrain, ytrain)
print('Train: ' , model.score(xtrain,ytrain))
print('Test: ' , model.score(xtest,ytest))
plt.title('Árboles de decision')
```

```
#Crear y entrenar modelo
from sklearn.svm import SVR
model = SVR(kernel = 'rbf', gamma=1, C = 1)#linear poli irvf
model.fit(xtrain, ytrain)
print('Train: ' , model.score(xtrain,ytrain))
print('Test: ' , model.score(xtest,ytest))
plt.title('SVR')
```

```
#Crear y entrenar modelo
from sklearn.neural_network import MLPRegressor
model = MLPRegressor(hidden_layer_sizes=(500,20), solver = 'adam', activation = 'tanh' , batch_size=20)
model.fit(xtrain, ytrain)
print('Train: ' , model.score(xtrain,ytrain))
print('Test: ' , model.score(xtest,ytest))
plt.title('MLPRegressor')
```

```
#Crear y entrenar modelo
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=10, weights='uniform')
model.fit(xtrain, ytrain)
print('Train: ' , model.score(xtrain,ytrain))
print('Test: ' , model.score(xtest,ytest))
plt.title('KNN')
```

```
#Crear y entrenar modelo
from sklearn.kernel_ridge import KernelRidge
model = KernelRidge(kernel = 'rbf' , alpha = 0.1)#linear poli irvf
model.fit(xtrain, ytrain)
print('Train: ' , model.score(xtrain,ytrain))
print('Test: ' , model.score(xtest,ytest))
plt.title('KernelRidge')
```

```
#Dibujar
xnew = np.linspace(-3, 3, 200).reshape(-1, 1)
ynew = model.predict(xnew)
plt.plot(xnew, ynew, '-k', linewidth = 3 )
plt.plot(xtrain,ytrain, '.b')
plt.plot(xtest,ytest, '.r')
plt.xlabel(r'$x$', fontsize=15)
plt.xlabel(r'$y$', fontsize=15)
plt.axis([-3,3,0,10])
plt.show()
```